We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

15

A New Mathematical Model for Flexible Flow Lines with Blocking Processor and Sequence-Dependent Setup Time

R. Tavakkoli-Moghaddam[1] and N. Safaei[2]
[1] *Department of Industrial Engineering, Faculty of Engineering, University of Tehran,*
[2] *Department of Mechanical and Industrial Engineering, University of Toronto,*
[1] *Iran,* [2] *Canada*

1. Introduction

This chapter presents a novel, mixed-integer programming model of the flexible flow line problem that minimizes the makespan of a product. The proposed model considers two main constraints, namely blocking processors and sequence-dependent setup time between jobs. We extend two previous studies conducted by Kurz and Askin (2004) and Sawik (2001), which considered only one of the foregoing constraints. However, this chapter considers both constraints jointly for flexible flow lines. A flexible flow line consists of several parallel processing stages in series, separated by finite intermediate buffers, in which each stage has one or more identical parallel processors. The line produces several different jobs, and each job must be processed by at most one processor at each stage. The completed job may remain on a machine and block the processor until a downstream processor becomes available for processing in the next stage; this is known as the *blocking processor* constraint. In the *sequence-dependent setup time* constraint , the processing of each job requires a setup time for preparing the processor that is immediately dependent on the preceding job. The objective, therefore, is to determine a production schedule for all jobs in such a way that they are completed in a minimum period of time (i.e., makespan). A number of numerical examples are solved and some computational results are reported to verify the performance of the proposed model. Finally, areas for future research are identified.

A flexible flow line consists of several processing stages in series, separated by finite inter-stage buffers, where each stage includes one or more identical parallel machines. The line produces several different job types. Each job must be processed by at most one machine in each stage. A processed job on a machine in some stage is transferred either directly to an available machine in the next stage (or another downstream stage depending on the job-processing route), or, when no intermediate buffer storage is available, to a buffer ahead of that stage. The job may remain on the machine and block it until a downstream machine becomes available (i.e., a *blocking processor*) (McCormick, 1989; Hall and Sriskandarajah, 1996; Sawik, 2000; Sawik, 2002). However, this blockage prevents another job from being processed on the blocked machine. Actually, a flexible flow line represents a special type of traditional flow shop, in which there is only one machine in each stage and unlimited intermediate storage between successive machines. The flexible flow line with unlimited

Source: Multiprocessor Scheduling: Theory and Applications, Book edited by Eugene Levner,
ISBN 978-3-902613-02-8, pp.436, December 2007, Itech Education and Publishing, Vienna, Austria

intermediate buffers has been also referred to a *hybrid flow line* (Blazewicz et al., 1994). Blocking scheduling problems arise in modern manufacturing environments, such as just-in-time production systems or flexible assembly lines, that have limited intermediate buffers between machines, or no buffers (e.g., surface mount technology (SMT) lines in the electronics industry for assembling printed circuit boards (Sawik, 2001)).

Setup includes work required to prepare the machine, process, or bench for job parts or the cycle. This presentation includes obtaining tools, positioning work-in-process inventory, returning tools and fixtures, cleaning up, setting the required jigs and fixtures, adjusting tools, and inspecting materials. Because of its complexity, in most studies, the setup operation (time and/or cost) has been considered negligible and hence ignored, or considered as part of the processing time in the case of setup times. While this may be justified for some scheduling problems, many other situations call for explicit (separable) setup time consideration. For a separable setup, two types of problem exist. In the first type, setup depends only on the job to be processed, hence is called *sequence-independent*. In the second type, setup depends on both the job to be processed and the immediately preceding job, hence is called *sequence-dependent* (Allahverdi et al., 1999).

2. Literature Review

The literature on the traditional flow shop and parallel machines scheduling is abundant and contains various optimization and approximation algorithms (Blazewicz et al., 1994). In addition, scheduling for flexible lines has been analyzed extensively in the literature over the last three decades. Kusiak (1988) considered flexible machining and assembly systems as two dependent subsystems, and proposed a heuristic two-level scheduling algorithm for a system consisting of a machining and an assembly subsystem in a flexible manufacturing system (FMS). Brandimart (1993) proposed a hierarchical algorithm for the flexible job shop scheduling problem based on a tabu search algorithm to minimize the makespan and the total weighted tardiness. Daniels and Mazzola (1993) used a tabu search algorithm for the flexible-resource flow shop scheduling problem (FRFSP). They introduced the FRFSP as a generalization of the flow shop scheduling problem. It explicitly considers the dynamic allocation of a flexible resource to machines, with operation processing times determined as a function of the amount of assigned resource. This problem requires that job-sequencing and resource-allocation decisions be made in conjunction, thus creating an environment in which significant operational benefits can be realized. Control of operation processing times by means of strategic resource allocation is a familiar concept in the project management literature. Riezebos et al., (1995) introduced a special instance of the flow shop scheduling problem originating from flexible manufacturing systems. In this problem, there is one machine at each stage. A job may require multiple operations at each stage. The first operation of a job on stage j cannot start until the last operation of the job on stage j-1 has finished. Preemption of the operations of a job is not allowed. To move from one operation of a job to another requires a finite amount of time, called a time lag. This time lag is independent of the sequence and may not be the same for all operations or jobs. During a time lag of a job, operations of other jobs may be processed.

Lee and Vairaktarakis (1998) compared the throughput performance of several flexible flow shop and job shop designs. They considered the two-stage assembly flow shops with m parallel machines in Stage 1 and a single assembly facility in Stage 2. Every upstream operation can be processed by any one of the machines in Stage 1 prior to the assembly

stage. They also studied a similar design where every Stage 1 operation is processed by a predetermined machine. For both designs, they presented heuristic algorithms with good worst-case error bounds, and showed that the average performance of these algorithms is near optimal. Jayamohan and Rajendran (2000) investigated the effectiveness of two approaches using different dynamic dispatching rules for the scheduling of flexible flow shops minimizing the flow times and tardiness of the jobs. Quadt and Kuhn (2005) considered a lot-sizing and scheduling problem of flexible flow lines for a semiconductor industry that minimizes the mean flow time as well as set-up, inventory holding, and back-order costs. Hong et. al., (2005) introduced a new fuzzy flexible flow shops for more than two machine centers with uncertain processing times and triangular membership functions. They also applied the triangular fuzzy LPT algorithm to allocate jobs and triangular fuzzy Palmer algorithm to find suitable sequence for the jobs. Alisantoso et al., (2003) proposed an immune algorithm for the scheduling of a flexible flow shop for PCB manufacturing. Torabi et al., (2005) studied the common cycle multi-product lot-scheduling problem in deterministic flexible job shops, and proposed an efficient enumeration method to determine the optimal solution for their model. Tavakkoli-Moghaddam and Safaei, (2005) proposed a queen-bee-based genetic algorithm to schedule flexible flow lines while considering the blocking processor. Tavakkoli-Moghaddam et al., (2007) also proposed a memetic algorithm to solve the mentioned scheduling problem. Jungwattanakit et al., (2007) formulated a 0–1 mixed-integer program to address the flexible flow shop scheduling problem in the textile industries that determines a schedule by minimizing a convex combination of makespan and the number of tardy jobs.

Research on the development of scheduling algorithms for flexible flow lines with finite or limited capacity buffers, or with no in-process buffers, is mostly restricted to the heuristics domain, in which good solutions are produced in reasonable computing times (Sawik, 1993; Sawik, 1994). Sawik (2000; 2001; 2002) first proposed an integer programming formulation for scheduling flexible flow lines with blocking processor and limited buffers. Sawik (2001) presented new mixed-integer programming formulations for blocking scheduling of SMT lines for printed wiring board assembly to minimize the makespan. He tested the model for small-sized problems (e.g., five stages and ten jobs). Kaczmarczyk et al., (2004) proposed a new mixed integer programming formulation for general or batch scheduling in SMT lines with continuous or limited machine availability. Their formulation is an improved version of the model presented by Sawik (2001), incorporating new cutting constraints on decision variables. They also presented a new formulation for batch scheduling with various specific cutting constraints. Tavakkoli-Moghaddam and Safaei (2006) presented an intial idea to consider both the blocking processor and sequence dependent setup time in flexible flow lines. Kis and Pesch (2005) provided a comprehensive and uniform overview on exact solution methods for flexible flow shops with branching, bounding, and propagating of constraints, under the following two objective functions: minimizing both the makespan and mean flow time of a schedule. Quadt and Kuhn (2007) also presented a taxonomy for flexible flow line scheduling procedures that included heuristic, metaheuristic, and holistic approaches.

The significance of setup times has been investigated in several studies. Wilbrecht and Prescott (1969) found that sequence-dependent setup times were significant when a job shop was operated at or near full capacity. In a survey of industrial managers, Panwalkar et al., (1973) discovered that out of about three-quarters of the managers' reports, at least some of

their scheduled operations require sequence-dependent setup times, while approximately 15% reported all operations requiring sequence-dependent setup times. Flynn (1986) determined that applications of both sequence-dependent setup procedures and group technology principles increased output capacity in a cellular manufacturing shop. Wortman (1992) also underlined the importance of considering sequence-dependent setup times for the effective management of manufacturing capacity. Krajewski et al., (1987) examined those factors in a production environment that had the biggest influence on performance and concluded that, regardless of the production system in use, simultaneous reduction of setup times and lot sizes was the most effective way to reduce inventory levels and improve customer service. Kurz and Askin (2004) presented an integer programming (IP) approach for a flexible flow line problem with infinite buffer and sequence-dependent setup time; their model does not consider blocking processor. A major disadvantage of the above integer programming approaches to scheduling is the need for solving large mixed-integer programs to obtain meaningful optimal solutions (Greene and Sadowski, 1986; Jiang and Hsiao, 1994). The size and complexity of the integer programming formulation increase when introduction of finite-capacity buffers results in a blocking scheduling problem. Although recent theoretical advances in integer programming and the advent of sophisticated computer hardware have enabled very powerful commercial software packages to come into use, large-sized problems cannot be optimaly solved within a reasonable time. Thus, heuristic or metaheuristic algorithms must be used for solving large and complex problems (Kurz and Askin, 2004).

While recent advances in manufacturing technologies such as flexible manufacturing systems (FMSs) or single-minute exchange of die (SMED) concepts have reduced the influence of setup time, there are still many environments where setup time is significant. There are also many practical applications that support separate consideration of setup tasks from processing tasks. These applications can be found in various shop types and environments; e.g., production, service, and information processing. Pinedo (1995) described a paper-bag factory where setup was needed when the machine was switched between types of paper bags, and the setup duration depended on the degree of similarity between consecutive batches, e.g., size and number of colors. The printing industry provides numerous applications of sequence-dependent setups where the machine cleaning involved depends on the color of the current and immediate following orders (Conway et al., 1967). In several textile industry applications, setup for weaving and dying operations depends on the job sequence. In the container and bottle industry, the settings change depending on the sizes and shapes of the containers. Further, in the plastic industry, different types and colors of jobs require sequence-dependent setups (Das et al., 1995; Franca, 1996; Srikar and Ghosh, 1986; Bianco, 1988). Similar practical situations arise in the chemical, pharmaceutical, food processing, metal processing, and paper industries (Bitran and Gilbert, 1990). Also, in an automatic turning center (ATC), setup time depends on the difference in the number and types of tools currently mounted on the turret and those required for the next work piece. Other examples of sequence-dependent setup time applications include a semiconductor testing facility (Kim and Bobrowski, 1994) and a machine shop environment (Ovacik and Uzsoy, 1992). Sule and Huang (1983) described the activities typically associated with sequence-dependent and sequence-independent operations in machine shop environments. Allahverdi et al., (1999) conducted a comprehensive review of setup-time research for scheduling problems classifying into batch, non-batch, sequence-independent, and

sequence-dependent setup. Also, Wang (2005) reviewed research on flexible flow shops (FFSs). Botta-Genoulaz (2000) solved a FFS problem with precedence constraints, time lags, setup and removal times, and due dates to minimize the maximum lateness.

In this chapter, we consider the flexible flow line problem (FFLP) with sequence-dependent setup time, without intermediate buffers that may lead to blocking processors. Simultaneouse consideration of both sequence-dependent setup time and blocking processor make the problem very complex for modeling and solving. We present a mixed-integer programming model that is optimally solved by a branch-and-bound (B/B) approach for small-sized problems. The rest of the chapter is organized as follows. The problem is described in Section 3; the proposed model is presented in Section 4; computational results are reported in Section 5; and in Section 6, conclusions are presented.

3. FFLP with Sequence-Dependent Setup Time and Blocking Processor

As mentioned earlier, the flexible flow lines problem (FFLP) with blocking (FFLPB) processor is a flexible flow line scheduling problen with no intermediate buffers or in-process buffers (Sawik, 2000). A processed job on a machine may remain there and block the processor until a downstream processor becomes available for processing in the next stage. A unified modeling approach is adopted with the buffers viewed as machines with zero processing times. As a result, the scheduling problem with buffers can be converted into one with no buffers but with blocking (Sawik, 1993 and 1995). The blocking time of a machine with zero processing time denotes job *waiting time* in the buffer represented by that machine. We assume that each job must be processed in all stages, including the buffers stages. However, zero blocking time in a buffer stage indicates that the corresponding job does not need to wait in the buffer. It is worth noting that for each buffer stage, job completion time is equal to its departure time from the previous stage, since the processing time is zero. In the notation proposed by Sawik (2000), buffers and machines are jointly called *processors*.

In this chapter, the FFLB problem consists of m processing stages in series, as shown in Figure 1. Each stage i (i =1,…, m) is made up of $n_i \geq 1$ identical parallel processors. The system produces K jobs of various types. Each job must be processed without preemption on exactly one processor in each of the stages sequentially. That is, each job must be processed in stage 1 through stage m, in that order. The order of processing the jobs in every stage is identical and determined by an input sequence in which the jobs enter the line. Let p_{ik} be the processing time for job k (k =1,…,K) in stage i. Also, the completion time for job k in stage i is denoted by c_{ik} , and d_{ik} is its departure time from stage i. Processing without preemption indicates that job k completed in stage i at time c_{ik} had started its processing in that stage at time c_{ik} - p_{ik} . Job k completed in stage i at time c_{ik} departs at time $d_{ik} \geq c_{ik}$ to an available processor in the next stage i+1. If time c_{ik} of all n_{i+1} processors in stage i+1 are occupied, then the processor in stage i is blocked by job k until time $d_{ik} = c_{(i+1)k}$ - $p_{(i+1)k}$, when job k starts processing on an available processor in stage i+1 (see Figure 2). Note that $c_{(i+1)k}$ is determined with respect to $c_{(i+1)(k-1)}$. The objective is to determine an assignment of jobs to processors in each stage over a scheduling horizon in such a way that all the jobs are completed in a minimum time in order to minimize the makespan (i.e., $C_{max} = \max_k\{c_{mk}\}$).

With blocking processor, on the other hand, it is possible that we encounter idle time for processors. In Figure 3, job l must be processed on stage i+1 (on the same processor) immediately before job k, where $c_{ik} > d_{(i+1)l}$. Therefore, the corresponding processor incurs an idle time in interval ($d_{(i+1)l}$, c_{ik}]. As depicted in Figure 3, the complete and departure times for

job k in stage i are the same, because the corresponding processor in stage i+1 is idle at the same time and job l can be processed on stage i+1 immediately after completion in stage i.

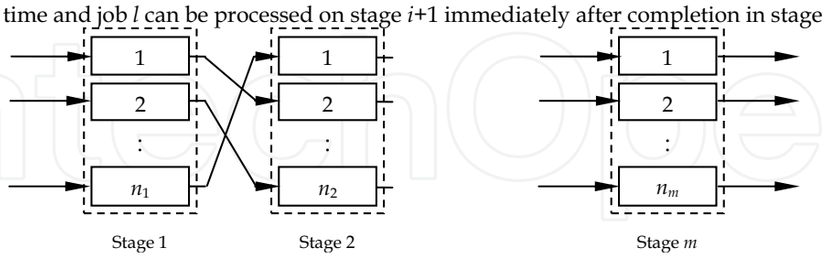


Figure 1. A flexible flow line with no intermediate buffers

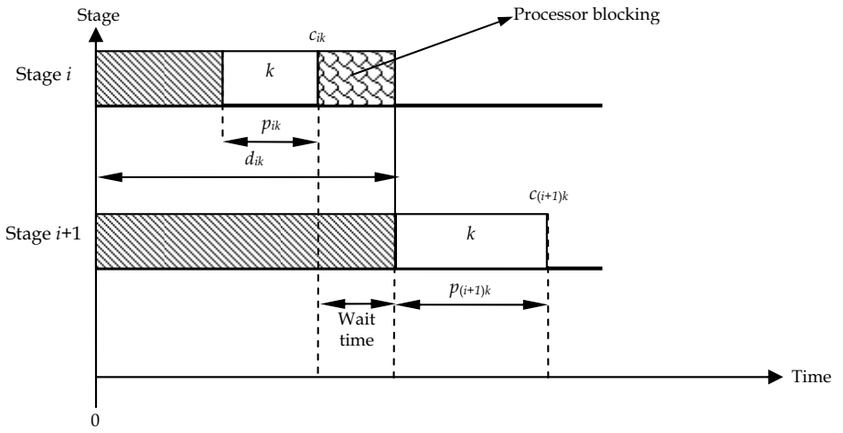


Figure 2. A schema of processor blocking

As noted earlier, setup time can include the time for preparing the machine or the processor. In an FFLPB with sequence-dependent setup time (FFLPB-SDST), it is assumed that the setup time depends on both jobs to be processed, the immediately preceding job, and the corresponding stage. Thus, a proper operation sequence on the processors has a significant effect on the makespan (i.e., C_{max}). As already assumed, the processors in each stage are identical, whereas the stages are different. Therefore, it is assumed that the setup time also depends on the stage type. A schema of sequence-dependent setup time in the FFLPB is illustrated in Figure 4. Job q must be processed immediately before job k in stage i. Also, job l must be processed immediately before job k in stage i+1. s_{iqk} is equal to the processor setup time for job k if job q is the immediately preceding job in the sequence operation on the corresponding processor. Likewise, $s_{(i+1)lk}$ is equal to the processor setup time for job k if job l is the immediately preceding job. Job q is completed in stage i at time c_{iq} and departs as time $d_{iq} \geq c_{iq}$ to an available processor in stage i+1 (excepting the one that is processing job k). As a result, job k is started at time $d_{iq}+s_{iqk}$ in stage i and departs at time $d_{ik} \geq d_{(i+1)l}$ to stage i+1. Likewise, job l is completed in stage i+1 at time $c_{(i+1)l}$ and departs at time $d_{(i+1)l} \geq c_{(i+1)l}$ to an available processor in the next stage. As a result, job k started at time $d_{(i+1)l} + s_{(i+1)lk}$ in stage

i+1 and completed at time $c_{(i+1)k}$. It is worth noting that the blocking processor or idle times cannot be used as setup time, because we assume the preparing processor requires the presence of a job.

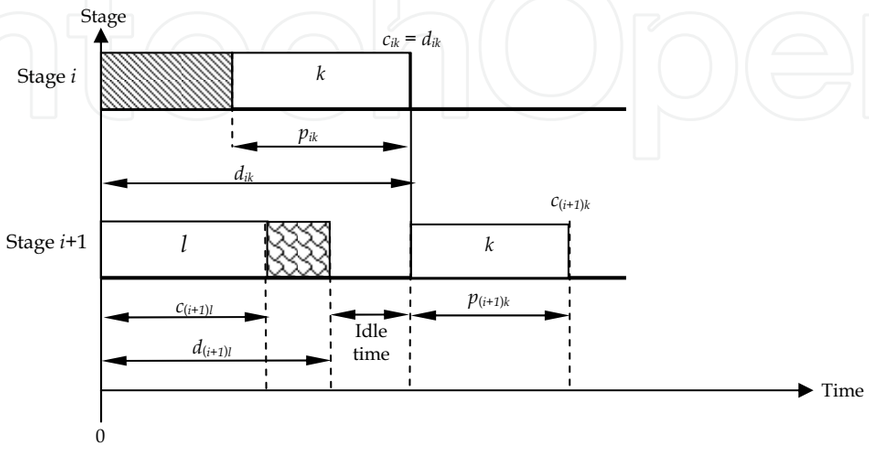


Figure 3. A schema of idle time

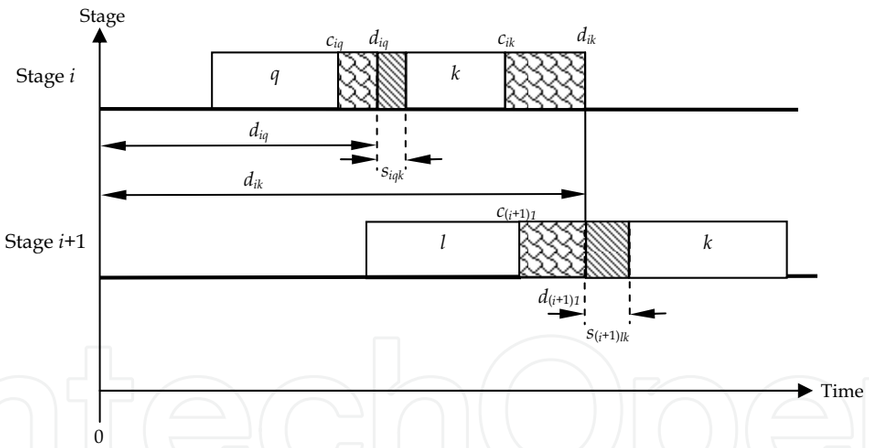


Figure 4. A schema of sequence-dependent setup time in FFLPB

4. Problem Formulation

In this section, we present a proposed model for the FFLP by considering both the blocking processor and sequence-dependent setup time. This model belongs to the mixed-integer nonlinear programming (MINLP) category. Then, we present a linear form for the proposed model. Without loss of generality, the FFLP can be modeled based on a traveling salesman

problem approach (TSP), since each processor at each stage plays the role of salesman once jobs (nodes) have been assigned to the processor. In this case, the sum of setup time and processing time indicates the distance between nodes. Thus, essentially the FFLP is an NP-hard problem (Kurz and Askin, 2004). A detailed breakdown of the proposed model follows.

4.1. Assumptions

The problem is formulated under the following assumptions. Like Kurz and Askin (2004), we also consider blocking processor and sequence-dependent setup times.

1. Machines are available at all times, with no breakdowns or scheduled or unscheduled maintenance.
2. Jobs are always processed without error.
3. Job processing cannot be interrupted (i.e., no preemption is allowed) and jobs have no associated priority values.
4. There is no buffer between stages, and processors can be blocked.
5. There is no travel time between stages; jobs are available for processing at a stage immediately after departing at previous stage.
6. The ready time for all jobs is zero.
7. Machines in parallel are identical in capability and processing rate.
8. Non-anticipatory sequence-dependent setup times exist between jobs at each stage. After completing processing of one job and before beginning processing of the next job, some sort of setup must be performed.

4.2. Input Parameters

m = number of processing stage.
K = number of jobs.
n_i= number of parallel processors in stage i.
p_{ik} = processing time for job k in stage i.
s_{ilk} = processor setup time for job k if job l is the immediately preceding job in sequence operation on the processor i. As discussed earlier, we assume that processors at each stage are identical, thus S_{ilk} is independent of index j, i.e., the processor index.

4.3. Indices

i = processing stage, where i =1,…, m.
j = processor in stage, where j =1,…, n_i.
k, l = job, where k, l =1,…, K.

4.4. Decision Variables

C_{max} = makespan.
c_{ik} = completion time of job k at stage i.
d_{ik} = departure time of job k from stage i.
x_{ijlk} = 1, if job k is assigned to processor j in stage i where job l is its predecessor job; otherwise x_{ijlk} = 0. Two nominal jobs 0 and K+1 are considerd as the first and last jobs, respectively (Kurz and Askin, 2004). It is assumed that nominal jobs 0 and K+1 have zero setup and process time and must be processed on each processor in each stage.

4.5. Mathematical Formulation

Min C_{max}

s.t.

$$\sum_{j=1}^{n_i}\sum_{l=0,l\neq k}^{K} x_{ijlk} = 1 \qquad \forall i,k \tag{1}$$

$$\sum_{l=0,l\neq k}^{K} x_{ijlk} = \sum_{q=1,q\neq k}^{K+1} x_{ijkq} \qquad \forall i,j,k \tag{2}$$

$$c_{1k} \geq p_{1k} + \sum_{j=1}^{n_i} x_{1j0k} s_{i0k} \qquad \forall k \tag{3}$$

$$c_{ik} - c_{(i-1)k} \geq p_{ik} + \sum_{j=1}^{n_i}\sum_{l=0}^{K} x_{ijlk} s_{ilk} \qquad \forall i>1,k \tag{4}$$

$$c_{ik} \geq p_{ik} + \sum_{j=1}^{n_i}\sum_{l=0,l\neq k}^{K} x_{ijlk}\left(s_{ilk} + d_{il}\right) \qquad \forall i,\ k=1,\ldots,K+1 \tag{5}$$

$$c_{ik} = d_{(i-1)k} + p_{ik} + \sum_{j=1}^{n_i}\sum_{l=0,l\neq k}^{K} x_{ijlk} s_{ilk} \qquad \forall i>1,k \tag{6}$$

$$c_{mk} = d_{mk} \qquad \forall k \tag{7}$$

$$C_{max} \geq c_{mk} \qquad \forall k \tag{8}$$

$$x_{ijlk} \in \{0,1\} \quad \forall i,j,l,k \quad ; \quad c_{ik}, d_{ik} \geq 0 \quad \forall i,k$$

The objective function is to minimize the schedule length. Constraint (1) ensures that each job *k* in every stage is assigned to only one processor immediately after job *l*. Constraint (2), which is complementary to Constraint (1), is a flow balance constraint, guaranteeing that jobs are performed in well-defined sequences on each processor at each stage. This constraint determines which processors at each stage must be scheduled. Constraint (3) calculates the complete time for the first available job on each processor at stage 1. Likewise, Constraint (4) calculates the complete time for the first available job on each processor in other stages, and also guarantees that each job is processed in all downstream stages with regard to setup time related to both the job to be processed and the immediately preceding job. Constraint (5) controls the formation of the processor's blocking. Constraint (6) calculates the processing of a job depending on the processing of its predecessor on the same processor in a given stage. This constraint controls creating the processor's idle time. Both constraint sets (5) and (6) ensure that a job cannot begin setup until it is available (done at the previous stage) and the previous job at the current stage is complete. Constraint (6) indicates that the processing of each job in every stage starts immediately after its departure from the previous stage plus the setup time of the immediately preceding job. Actually, this constraint calculates the departure time related to each job at each stage except for the last stage. Constraint (7) ensures that each product leaves the line as soon as it is completed in the latest stage. Finally, Constraint (8) defines the maximum completion time.

4.6. Model Linearization

The proposed model has a nonlinear form because of the existence of Constraint (5). Thus, it cannot be solved optimally in a reasonable time by programming approaches. Thus, we present a linear form for the proposed model by defining the integer variable y_{ijlk} and changing Constraint (5), as indicated in the following expressions.

$$y_{ijlk} \geq \left(s_{ilk} + d_{il}\right) - M \times \left(1 - x_{ijlk}\right) \qquad \forall i, j, l, k \tag{9}$$

$$c_{ik} \geq p_{ik} + \sum_{j=1}^{n_i} \sum_{l=1, l \neq k}^{K} y_{ijlk} \qquad \forall i, k \tag{10}$$

where M is an arbitrary big number. Constraint (5) must be replaced by Constraints (9) and (10) in the above proposed model.

4.7 A Lower Bound for the Makespan

In this section, we develop a processor based on a lower bound and evaluate schedules produced in this manner with other heuristic (or metaheuristic) approaches. The proposed lower bound was developed based on the lower-bound method presented by Sawik (2001) for the FFLPB. The proposed lower bound resulted from the following theorem:

Theorem. Equation (11) is the lower bound on any feasible solution of the proposed model.

$$LB = \max_{i=1}^{m} \left\{ \sum_{k=1}^{K} \frac{\left(p_{ik} + S_{ik}\right)}{n_i} + \sum_{h=1}^{i-1} \min_{k=1}^{K} \left\{p_{hk} + S_{hk}\right\} + \sum_{h=i+1}^{m} \min_{k=1}^{K} \left\{p_{hk} + S_{hk}\right\} \right\} \tag{11}$$

where,

$$S_{ik} = \min_{l=1}^{K} \left\{s_{ilk}\right\} \qquad \forall i, k$$

Proof. Let S_{ik} be the minimum time required to set up job k at stage i. We know that every job k must be processed at each stage and must also be set up. In an optimistic case, we assume that the work-load incurred to processors at each stage is identical. Thus, each processor at stage i has the minimum mean workload $(1/n_i) \times (\sum_k [p_{ik} + S_{ik}])$ (i.e., the first term in Equation (11)). According to constraint sets (4) and (5), a job cannot begin setup until it is available and the previous job at the current stage is complete. Actually, constraint sets (4) and (5) remark two facts. First, each processor at each stage i incurs an idle time because of waiting for the first available job. A lower bound for this waiting time in stage i can be the second term in Equation (11). Second, each processor at each stage i incurs an idle time after accomplishment of processing untill the end of scheduling. This idle time is equal to the sum of the minimum time to processing jobs at the next stages (i.e., i+1, ..., m). A lower bound for this idle time can be the third term in Equation (11). The sum of the above three terms indicates a typical lower bound in terms of an optimistic scheduling in stage i. Thus, LB in Equation (11) is a lower bound on any feasible solution.□

5. Numerical Examples

In this section, many numerical examples are presented, and some computational results are reported to illustrate the efficiency of the proposed approach. Fourteen small-sized problems are considered in order to evaluate the proposed model. Each problem has some integer processing times selected from a uniform distribution between 50 and 70, and

integer setup times selected from a uniform distribution between 12 and 24 (Kurz and Askin, 2004). To verify the model and illustrate the approach, problems were generated in the following three categories: (1) Classical flow shop (one processor at each stage), termed CAT1 problems; (2) FFLP with the same number of processors at each stage, termed CAT2 problems; and (3) FFLP with a different number of processors at each stage, termed CAT3 problems. The CAT1 problems are considered simply to verify the performance of the proposed model. To make the comparison of runs simpler and also for standardization, we assume that the total number of processors in all stages is equal to double the number of stages, i.e., $\sum_k n_k = 2\times m$. For example, a problem with three stages has six processors in total. These problems have been solved by the Lingo 8.0 software on a personal computer with Celeron M 1.3 GHz CPU and 512 MB of memory. Each problem is allowed a maximum of 7200 seconds of CPU time (two hours) using the Lingo setting (→/Option/General Solver/time Limitation = 7200 Sec.).

Table 1 contains additional information about CAT1 problems for finding optimal solutions (i.e., classical flow shop). Problems are considered with two, three, and four stages and more than four jobs. The values for Columns 'B/B Steps' and 'CPU Time' are two vital criteria for measuring the severity and complexity of the proposed model. Also, the dimension of the problem is shown when regarding the number of 'Variables' and 'Constraints' in Table 1. In CAT1 problems, the number of variables is less than the number of constraints. Thus, CAT1 problems are more severe than CAT2 and CAT3 problems in terms of the time complexity and computational time required. For example, despite all efforts, a feasible solution is not found in 2 hours for problem 10 (i.e., 6 jobs and 4 stages = 4 processors). However, for problem 3 in Table 2 with nearly the same condition and dimension (i.e., 6 jobs and 2 stages = 4 processors), the optimal solution is reached in less than one hour. Likewise, for problem 3 in Table 3 (i.e., 6 jobs and 2 stages = 4 processors), the optimal solution is reached in less than three minutes. To illustrate the complexity of solving FFLPB-SDST, the behavior of the B/B's CPU time vs. increasing the number of jobs for different numbers of stages related to data provided in Table 1 is shown in Figure 5. As the figure indicates, by increasing the number of stages, the CPU time increases progressively. Table 1 also shows that increasing the number of stages (or processors) leads to a greater increase in computational time, rather than an increase in the number of jobs. Table 2 contains additional problem characteristics and information for optimal solutions related to CAT2 problems (i.e., there are two processors at each stage). Likewise, Table 3 contains additional problem information for obtaining optimal solutions related to CAT3 problems (i.e., different numbers of processors at each stage).

				Number of					
No.	*K*	*m*	n_i	Variables	Constraints	B/B Steps	CPU Time	C_{max}	LB
1	4	2	1,1	81	95	330	00:00:03	384	383
2	5	2	1,1	121	138	2743	00:00:17	450	445
3	6	2	1,1	169	189	151739	00:14:52	524	503
4	7	2	1,1	225	248	-	> 2 hours	610*	585
5	4	3	1,1,1	121	140	1849	00:00:25	465	430
6	5	3	1,1,1	181	204	9588	00:01:45	544	519
7	6	3	1,1,1	253	280	-	> 2 hours	615*	577
8	4	4	1,1,1,1	161	185	297	00:00:26	548	520
9	5	4	1,1,1,1	241	270	122412	00:21:20	627	605
10	6	4	1,1,1,1	337	371	-	> 2 hours	Infeasible**	700

* The best feasible objective value is found so far.
** A feasible solution is not found so far.

Table 1. Optimal solutions for CAT1 problems

No.	K	m	n_i	Number of Variables	Constraints	B/B Steps	CPU Time	C_{max}	LB
1	4	2	2,2	145	145	872	00:00:05	230	218
2	5	2	2,2	221	220	10814	00:00:55	295	252
3	6	2	2,2	313	311	240586	00:47:36	299	291
4	7	2	2,2	421	418	-	> 2 hours	380*	335
5	4	3	2,2,2	217	215	6644	00:00:32	314	306
6	5	3	2,2,2	331	327	232987	01:02:38	376	328
7	6	3	2,2,2	469	463	-	> 2 hours	389*	376
8	4	4	2,2,2,2	289	285	28495	00:02:23	395	392
9	5	4	2,2,2,2	441	434	-	> 2 hours	446*	390

* The best feasible objective value is found so far.

Table 2. Optimal solution for CAT2 problems

No.	K	m	n_i	Number of Variables	Constraints	B/B Steps	CPU Time	C_{max}	LB
1	4	2	1,3	209	145	251	00:00:03	381	380
2	5	2	1,3	321	220	1849	00:00:36	434	429
3	6	2	1,3	457	311	6921	00:02:32	527	523
4	7	2	1,3	617	418	-	> 2 hours	584*	577
5	4	3	2,1,3	311	215	754	00:00:11	437	426
6	5	3	2,1,3	481	327	89714	00:13:52	484	479
7	6	3	2,1,3	685	463	84304	00:25:26	574	570
8	7	3	2,1,3	925	623	-	> 2 hours	645*	639

* The best feasible objective value is found so far.

Table 3. Optimal solution for CAT3 problems

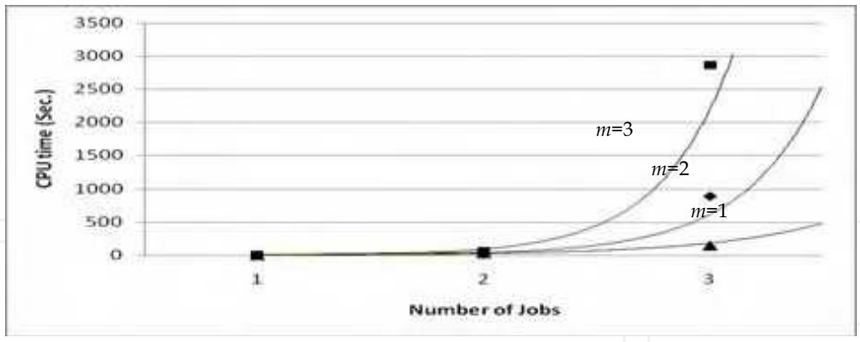


Figure 5. The behavior of the B/B's CPU time vs. increasing the number of jobs for a different number of stages

A linear regression analysis was made to fit a line through a set of observations related to values of C_{max} (i.e., makespan) vs. the lower bound (LB). Original figures were obtained from the results in Tables 1, 2, and 3. This analysis can be useful for estimating the C_{max} value for the large-sized problems genererated by using the form presented in this chapter.

A scatter diagram of C_{max} vs. the LB is shown in Figure 5. Obviously, the linear trend of the scatter diagram is noticeable. Table 4 contains regression results. According to Table 4, C_{max} can be estimated as $\hat{C}_{max} = 0.9833 \times LB + 0.0325$. The R^2 value, which is called the coefficient of determination, compares estimated with actual C_{max} values ranging from 0 to 1. If R^2 is 1, then there is a perfect correlation in the sample and there is no difference between the estimated C_{max} value and the actual C_{max} value. At the other extreme, if R^2 is 0, the regression equation is not helpful in predicting a C_{max} value. Thus, R^2 = 0.981 implies the goodness of fitness and observations. For instance, we generate a problem with 20 jobs and 3 stages belonging to CAT2 (two processors at each stage) that cannot be solved optimally in a reasonable time. According to Equation (14), the lower bound for the generated problem is 886, thus, the estimated C_{max} is 893. If some other approach can achieve a solution with a C_{max} value in the interval (886, 893], we can say that this is an efficient approach. Thus, interval (LB, $\hat{C}_{max}$] can be a proper criterion for evaluating the performance of other approaches.

Slop	Constant	R^2	Regression sum of squares	Residual sum of squares
0.9833	0.0325	0.981	912.44	202313.79

Table 4. Regression results

As further illustrations, we present a typical optimal scheduling for each category of problem, i.e., CAT1, CAT2, and CAT3, in Figures 7, 8, and 9, respectively. These figures are created by using the notations shown in Figure 6. Figure 7 illustrates the optimal scheduling for problem 9 in Table 1. For instance, there is a blocking time in stage 2 (S2-P2), that is close to the completion time of job 3, since job 2 is not departed from stage 3. In addition, there is a blocking processor and immediate idle time in stage 3 that is close to the completion time of job 3, because job 2 is not still departed from stage 4 and the completion time of job 1 in stage 2 is greater than departure time of job 3 in stage 3. It is worth noting that the processing sequence is the same at all stages implying a classical flow shop. Figure 8 depicts the optimal scheduling for problem 6 shown in Table 2, in which there is one tiny blocking time and several relatively long idle times. For instance, there is a tiny blocking time next to job 3 in stage 2 on processor 1 (S2-P1) because job 2 is not yet departed from stage 3 on processor 2 (S3-P2). Figure 8 also presents the processing sequence between each pair of observed jobs. For example, the departure time of job 2 is always later than the setup time (processing start time) of job 1 at the stages. In general, we expect few blocking times for CAT1 and CAT2 problems because there are an equal number of processors at each stage and the model endeavors to allocate the same workload to each processor at each stage for minimizing C_{max}. On the other hand, in CAT3 problems, we expect more blocking time because of the unequal number processor times at each stage. For instance, as shown in Figure 9, there are two relatively long blocking times in stage 1 because all jobs must be processed in stage 2 on only one processor. On the other hand, there are several relatively long idle times in stage 3 because of the above reason. Actually, stage 2 plays the role of bottleneck here.

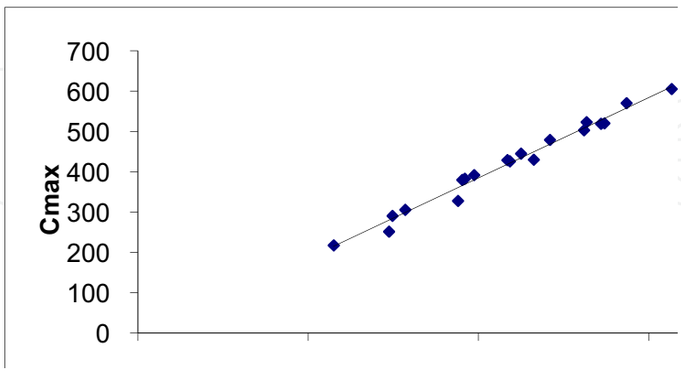


Figure 5. C_{max} vs. the lower bound (LB)

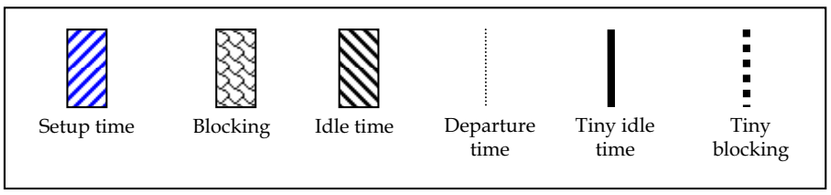


Figure 6. Legends

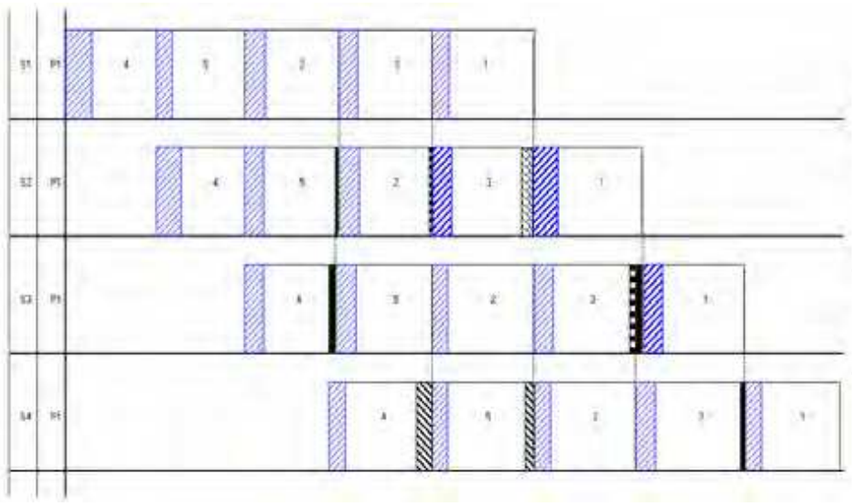


Figure 7. Optimal scheduling for problem 9 shown in Table 1 from CAT1

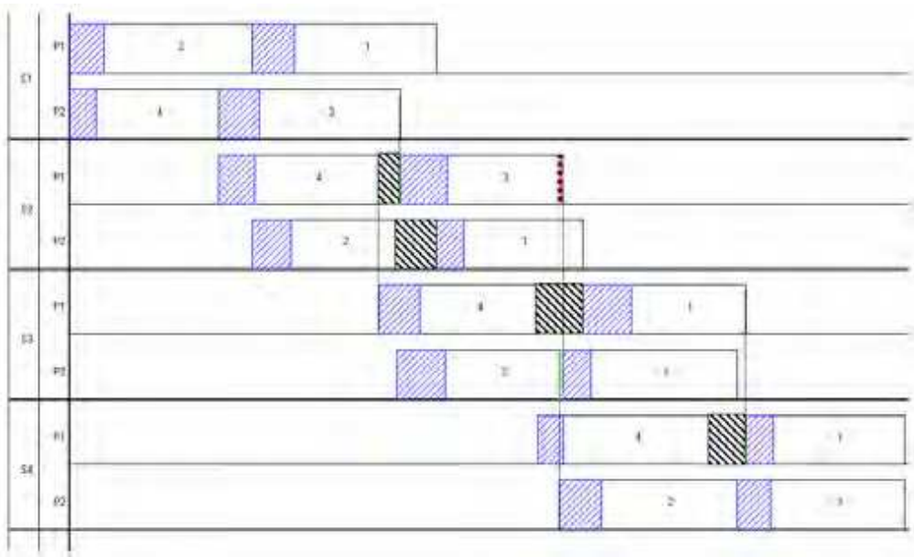


Figure 8. Optimal scheduling for problem 6 shown in Table 2 from CAT2

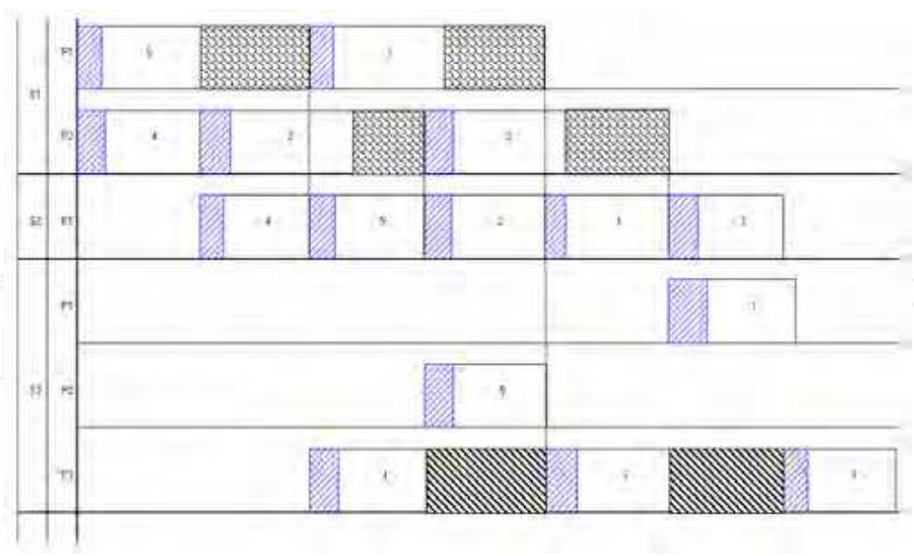


Figure 9. Optimal scheduling for problem 6 shown in Table 2 from CAT3

6. Conclusions

In this chapter, we presented a new mixed-integer programming approach to the flexible flow line problem without intermediate buffers by assuming in-process buffers and sequence-dependent setup time. The proposed mathematical model can provide an optimal schedule by considering blocking processor and idle time as well as sequence-dependent setup time. We solved the proposed model for three problem categories, i.e., classical flow shop (CAT1), stages with an equal number of processors (CAT2), and stages with an unequal number of processors (CAT3). Computation results showed that solving CAT3 problems requires low computational time, since there are less complex than CAT1 and CAT2 problems. On the other hand, in the classical flow shop case (i.e., CAT1), a high computational time is required. In many practical situations, the proposed model cannot optimally solve more than seven jobs with three stages (or six processors). Further, we developed a lower bound to evaluate the schedules produced with other heuristic or metaheuristic approaches. Also, a linear regression analysis was made to find a logical relationship between the makespan and its lower bound, which can be used in future research. The proposed model can be solved by other heuristic or metaheuristic approaches as well, and with uncertain processing times and/or setup times. It can also be solved using limited intermediate buffers instead of in-process buffers.

7. References

Alisantoso, D.; Khoo, L.P. & Jiang, P.Y. (2003). An immune algorithm approach to the scheduling of a flexible PCB flow shop. *Int. J. of Advanced Manufacturing Technology*, Vol. 22, pp. 819-827.

Allahverdi, A.; Gupta, J. & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega*, *Int. J. Mgmt Sci.*, Vol. 27, pp. 219-239.

Blazewicz, J.; Ecker, K.H.; Schmidt, G. & Weglarz, J. (1994). *Scheduling in Computer and Manufacturing Systems*. Berlin: Springer-Verlag.

Bianco, L.; Ricciardelli; S.; Rinaldi, G. & Sassano, A. (1988). Scheduling tasks with sequence-dependent processing times. *Naval Res. Logist*, Vol. 35, pp. 177-84.

Bitran, G.R. & Gilbert, S.M. (1990). Sequencing production on parallel machines with two magnitudes of sequence-dependent setup cost. *J. Manufact Oper Manage*, Vol. 3, pp. 24-52.

Botta-Genoulaz, V. (2000). Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. *Int. J. of Production Economics*, Vol. 64, Nos. 1–3, pp. 101–111.

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, Vol. 41, pp. 157-183.

Conway, R.W.; Maxwell, W.L. & Miller, L.W. (1967). *Theory of Scheduling*, Addison Wesley, MA.

Daniels, R.L. & Mazzola, J.B. (1993). A tabu-search heuristic for the flexible-resource flow shop scheduling problem. *Annals of Operations Research*, Vol. 41, pp. 207-230.

Das, S.R.; Gupta, J.N.D. & Khumawala, B.M. (1995). A saving index heuristic algorithm for flowshop scheduling with sequence dependent setup times. *J. Oper Res Soc*, Vol. 46, pp. 1365-73.

Franca, P.M.; Gendreau, M.; Laporte, G. & Muller, F.M. (1996). A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *Int J. Prod Econ*, Vol. 43, pp. 79-89.

Flynn, B.B. (1987). The effects of setup time on output capacity in cellular manufacturing. *Int. J. Prod Res*, Vol. 25, pp. 1761-72.

Greene, J.T. & Sadowski, P.R. (1986). A mixed integer program for loading and scheduling multiple flexible manufacturing cells. *European Journal of Operation Research*, Vol. 24, pp. 379-386.

Hall, N.G. & Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research*, Vol. 44, pp. 510-525.

Hong, T.-P.; Wang, T.-T. & Wang, S.-L. (2001). A palmer-based continuous fuzzy flexible flow-shop scheduling algorithm. *Soft Computing*, Vol. 6., pp. 426-433.

Jayamohan, M.S. & Rajendran, C. (2000). A comparative analysis of two different approaches to scheduling in flexible flow shops. *Production Planning & Control*, 2000, Vol. 11, No. 6, pp. 572-580.

Jiang, J. & Hsiao, W. (1994). Mathematical programming for the scheduling problem with alterative process plan in FMS. *Computers and Industrial Engineering*, Vol. 27, No. 10, pp. 5-18.

Jungwattanakit, J.; Reodecha, M.; Chaovalitwongse, P. & Werner, F. (2007). Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *Int. J. of Advanced Manufacturing Technology*, DOI 10.1007/s00170-007-0977-0.

Kaczmarczyk, W., Sawik, T., Schaller, A. and Tirpak T.M. (2004). Optimal versus heuristic scheduling of surface mount technology lines. *Int. J. of Production Research*, Vol. 42, No. 10, pp. 2083-2110.

Kim, S.C. & Bobrowski, P.M. (1994). Impact of sequence-dependent setup times on job shop scheduling performance. *Int. J. Prod Res.*, Vol. 32, pp. 1503-20.

Kis, T. & Pesch, E. (2005). A review of exact solution methods for the non-preemptive multiprocessor flowshop problem. *Eur. J. of Operational Research*, Vol. 164, No. 3, pp. 592-608.

Krajewski, L.J.; King, B.E.; Ritzman, L.P. & Wong, D.S. (1987). Kanban, MRP, and shaping the manufacturing environment. *Manage Sci*, Vol. 33, pp. 39-57.

Kurz, M.E. & Askin, R.G. (2004). Scheduling flexible flow lines with sequence-dependent setup times. *Eur. J. of Operational Research*, Vol. 159, pp. 66–82.

Kusiak, A. (1988). Scheduling flexible machining and assembly systems. *Annals of Operations Research*, Vol. 15, pp. 337-352.

Lee, C.-Y. & Vairaktarakis, G.L. (1998). Performance comparison of some classes of flexible flow shops and job shops. *Int. J. of Flexible Manufacturing Systems*, Vol. 10, pp. 379-405.

McCormick, S.T.; Pinedo, M.L.; Shenker, S. & Wolf, B. (1989). Sequencing in an assembly line with blocking to minimize cycle time. *Operation Research*, Vol. 37, pp. 925-936.

Ovacik, I.M. & Uzsoy, R.A. (1992). Shifting bottleneck algorithm for scheduling semiconductor testing operations. *J. Electron Manufact*, Vol. 2, pp. 119-34.

Panwalkar, S.S.; Dudek, R.A. & Smith, M.L. (1973). Sequencing research and the industrial scheduling problem. In: *Symposium on the Theory of Scheduling and its Applications*, Elmaghraby, S.E. (editor), pp. 29-38.

Pinedo, M. (1995), Scheduling: Theory, Algorithms, and Systems. Prentice Hall, NJ.

Quadt, D. & Kuhn, H. (2005). Conceptual framework for lot-sizing and scheduling of flexible flow lines. *Int. J. of Production Research*, Vol. 43, No. 11, pp. 2291-2308.

Quadt, D. & Kuhn, H. (2007). A taxonomy of flexible flow line scheduling procedures. *Eur. J. of Operational Research*, Vol. 178, pp. 686-698.

Riezebos, J.; Gaalman, G.J.C. & Gupta, J.N.D. (1995). Flow shop scheduling with multiple operations and time lags. *J. of Intelligent Manufacturing*, Vol. 6, pp. 105-115.

Sawik, T. (1993). A scheduling algorithm for flexible flow lines with limited intermediate buffers. *Applied Stochastic Models and Data Analysis*, Vol. 9, pp. 127-138.

Sawik, T. (1995). Scheduling flexible flow lines with no-process buffers. *Int. J. of Production Research*, Vol. 33, pp. 1359-1370.

Sawik, T. (2000). Mixed integer programming for scheduling flexible flow lines with limited intermediate buffers. *Mathematical and Computer Modeling*, Vol. 31, pp. 39-52.

Sawik, T. (2001). Mixed integer programming for scheduling surface mount technology lines. *Int. J. of Production Research, Vol.* 39, No. 14, pp. 2319- 3235.

Sawik, T. (2002). An Exact Approach for batch scheduling in flexible flow lines with limited intermediate buffers. *Mathematical and Computer Modeling*, Vol. 36, pp. 461-471.

Srikar, B.N. & Ghosh, S. (1986). A MILP model for the n-job, M-stage flowshop, with sequence dependent setup times. *Int. J. Prod Res.*, Vol. 24, pp. 1459-1472.

Sule, D.R. & Huang, K.Y. (1983). Sequence on two and three machines with setup, processing and removal times separated. *Int J Prod Res*, Vol. 21, pp. 723-32.

Tavakkoli-Moghaddam, R. & Safaei, N. (2005). A genetic algorithm based on queen bee for scheduling a flexible flow line with blocking, *Proceeding of the 1st Tehran International Congress on Manufacturing Engineering (TICME2005)*, Tehran: Iran, December 12-15, 2005.

Tavakkoli-Moghaddam, R.; Safaei, N. & Sassani, F., (2007). A memetic algorithm for the flexible flow line scheduling problem with processor blocking, *Computers and Operations Research*, Article in Press, DOI: 10.1016/j.cor.2007.10.011.

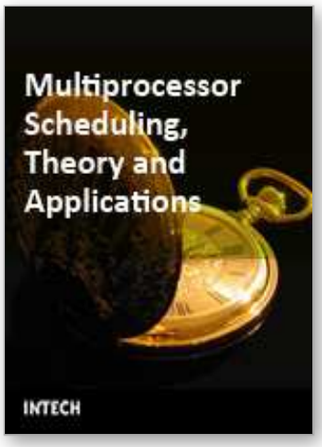
Tavakkoli-Moghaddam, R. & Safaei, N. (2006). Modeling flexible flow lines with blocking and sequence dependent setup time, *Proceeding of the 5th International Symposium on Intelligent Manufacturing Systems (IMS2006)*, pp. 149-158, Sakarya: Turkey, May 29-31, 2006.

Torabi, S.A.; Karimi, B. & Fatemi Ghomi, S.M.T. (2005). The common cycle economic lot scheduling in flexible job shops: The finite horizon case. *Int. J. of Production Economics*, Vol. 97, No. 1, pp. 52-65.

Wang, H. (2005). Flexible flow shop scheduling: optimum, heuristics and artificial intelligence solutions. *Expert Systems*, Vol. 22, No. 2, pp. 78-85.

Wilbrecht, J.K. & Prescott, W.B. (1969). The influence of setup time on job shop performance. *Manage Sci.*, Vol. 16, pp. B274-B280.

Wortman, D,B. (1992). Managing capacity: getting the most from your form's assets. *Ind. Eng*, Vol. 24, pp. 47-59.



Multiprocessor Scheduling, Theory and Applications

Edited by Eugene Levner

ISBN 978-3-902613-02-8

Hard cover, 436 pages

Publisher I-Tech Education and Publishing

Published online 01, December, 2007

Published in print edition December, 2007

A major goal of the book is to continue a good tradition - to bring together reputable researchers from different countries in order to provide a comprehensive coverage of advanced and modern topics in scheduling not yet reflected by other books. The virtual consortium of the authors has been created by using electronic exchanges; it comprises 50 authors from 18 different countries who have submitted 23 contributions to this collective product. In this sense, the volume can be added to a bookshelf with similar collective publications in scheduling, started by Coffman (1976) and successfully continued by Chretienne et al. (1995), Gutin and Punnen (2002), and Leung (2004). This volume contains four major parts that cover the following directions: the state of the art in theory and algorithms for classical and non-standard scheduling problems; new exact optimization algorithms, approximation algorithms with performance guarantees, heuristics and metaheuristics; novel models and approaches to scheduling; and, last but least, several real-life applications and case studies.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen